

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-274524

(43)Date of publication of application : 30.09.1994

(51)Int.Cl.

G06F 15/332

G06F 15/66

H03M 7/30

H04N 1/41

H04N 7/133

(21)Application number : 05-061613

(71)Applicant : HITACHI LTD

(22)Date of filing : 22.03.1993

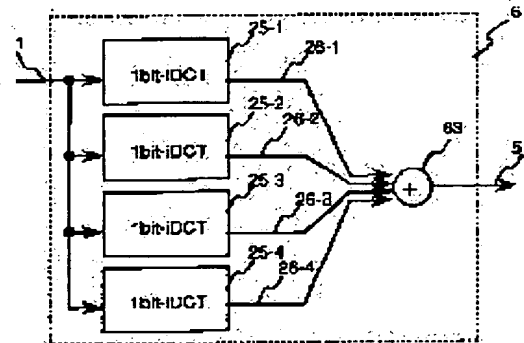
(72)Inventor : KIMURA JUNICHI

## (54) ORTHOGONAL TRANSFORMATION CIRCUIT AND INVERSE TRANSFORMATION CIRCUIT

(57)Abstract:

PURPOSE: To execute transformation or inverse transformation processings at an operation speed same as before without increasing a circuit scale by dividing data used for integrating into 1-bit data, replacing an integrating part with shift calculation and adding the data of respective bits after a product-sum operation.

CONSTITUTION: 1-bit iDCT circuits 25-1-25-4 are parallelly arranged and also an adder 63 for adding the arithmetic results 26-1-26-4 is arranged. That is, one of the 1-bit iDCT circuits 25-1-25-4 is allocated to each matrix, input signals 1 are calculated for each matrix broken down by each bit, the respective results 26-1-26-4 are added at the adder 63 and a final result is obtained. In this case, the 1-bit iDCT circuits 25-1-25-4 are circuits for integrating input vectors and the matrix for which the bit number of '1' included when the absolute values of the respective components of the matrix is binary displayed is equal to or less than 1 and are composed of shift circuits and adding-subtracting circuits.



## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平6-274524

(43)公開日 平成 6年(1994) 9月30日

(51)Int.Cl. <sup>5</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 15/332	S	7343-5L		
15/66	3 3 0 H	8420-5L		
H 0 3 M 7/30	A	8522-5 J		
H 0 4 N 1/41	B	9070-5C		
7/133	Z			

審査請求 未請求 請求項の数 6 O L (全 11 頁)

(21)出願番号 特願平5-61613

(22)出願日 平成 5年(1993) 3月22日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目 6番地

(72)発明者 木村 淳一

東京都国分寺市東恋ヶ窪 1丁目280番地

株式会社日立製作所中央研究所内

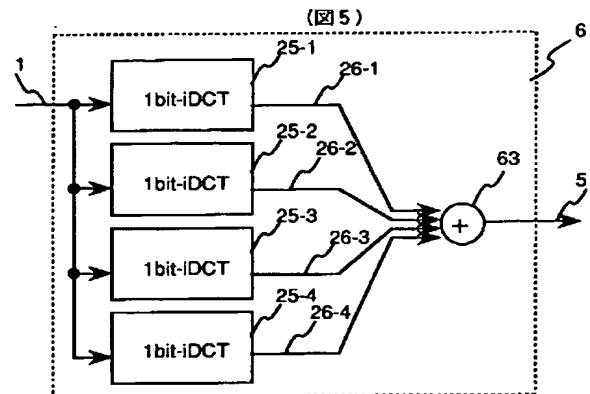
(74)代理人 弁理士 小川 勝男

(54)【発明の名称】 直交変換回路および逆変換回路

(57)【要約】

【構成】1ビット i D C T回路 2 5 - 1 ~ 4 を並列に配置し、それらの演算結果 2 6 - 1 ~ 4 を加算する加算器 6 3 より構成される。1ビット i D C T回路は行列の各要素の絶対値を2進数表示したときに含まれる“1”のビット数が一つ以下の行列と入力ベクトルの積算を行う回路であり、シフト回路、加減算回路からなる。

【効果】直交変換の行列演算を1ビット毎に計算することにより従来と同じ動作速度で、しかも、回路規模も増加させることなしに変換あるいは逆変換処理を実行することができる。



1

【特許請求の範囲】

【請求項1】N次のベクトルを入力し、入力ベクトルと直交行列の積算を行いN次の出力行列を得る直交変換回路において、直交行列に予め定めた係数値を掛け正規化する手段、正規化された行列の各要素の絶対値を2進数表示したときに含まれる“1”のビット数が一つ以下である行列の和あるいは差の形に分割する手段、分割された行列と入力ベクトルの積を計算する手段、各分割された行列との積算結果のベクトルを分割した手順に従って合成する手段、合成したベクトルに予め定められた出力係数を掛ける手段を具備することを特徴とする直交変換回路。

【請求項2】N次の直交変換されたベクトルを入力し、入力ベクトルと直交行列の積算を行いN次の出力行列を得る直交変換の逆変換回路において、直交行列に予め定めた係数値を掛け正規化する手段、正規化された行列の各要素の絶対値を2進数表示したときに含まれる“1”のビット数が一つ以下である行列の和あるいは差の形に分割する手段、分割された行列と入力ベクトルの積を計算する手段、各分割された行列との積算結果のベクトルを分割した手順に従って合成する手段、合成したベクトルに予め定められた出力係数を掛ける手段を具備することを特徴とする逆変換回路。

【請求項3】請求項1において、回路を縦続に接続し、 $N \times N$ 次の直交変換を行う直交変換回路。

【請求項4】請求項2において、回路を縦続に接続し、 $N \times N$ 次の逆変換を行う逆変換回路。

【請求項5】請求項1のベクトル合成において、分割し\*

$$F(u) = \frac{1}{2} C(u) \sum_{x=0}^7 f(x) \cos \left\{ \frac{(2x+1)u \times \pi}{16} \right\} \quad \dots \text{ (数1) }$$

【0004】 ※ ※ 【数2】

$$f(x) = \frac{1}{2} \sum_{u=0}^7 C(u) F(u) \cos \left\{ \frac{(2x+1)u \times \pi}{16} \right\} \quad \dots \text{ (数2) }$$

【0005】 ★ ★ 【数3】

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v=0 \\ 1 & \dots \text{ otherwise} \end{cases} \quad \dots \text{ (数3) }$$

【0006】数1のDCTおよび数2のiDCTは入力ベクトルとCOSで表現される $8 \times 8$ の行列との積によっても表現できる。数4ないし数8は数1のDCT、数2のiDCTを行列表現で表したものである。数8は数2のCOS部分、 $C(u)$ および $1/2$ の係数部分をすべてまとめたものである。数7の行列は正規直交行列であるため逆行列は元の行列の転置の形になり、DCTおよびiDCTの計算は同じ手順で計算する事ができる。

【0007】

【数4】

2

\*た数よりも少ないベクトルを用いて合成を行う手段を具備する直交変換回路。

【請求項6】請求項2のベクトル合成において、分割した数よりも少ないベクトルを用いて合成を行う手段を具備する逆変換回路。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は直交変換を用いた符号化あるいは復号化処理に係り、特に、画像信号の符号化、復号化のような大量データの高速な処理を可能とする回路に関する。

【0002】

【従来の技術】直交変換は予め定められた個数の信号を、同じ個数のそれぞれ直交するベクトルの成分に分解する手法である。直交変換の代表的なものに離散コサイン変換（以下DCT）がある。DCTについての詳細は、例えばケー アール ラオ(K.R.Rao)他著、安田浩／藤原洋共訳「画像符号化技術—DCTとその国際標準—(原著名: Discrete Cosine Transform Algorithms, Advantages, Applications)」(オーム社、1992.7)全編にわたって記されている。以下、DCTの説明を、本発明に関する部分のみ簡単に行う。数1および数3は8次のDCTの例である。DCTでは数1に示すようにCOSで表現されるベクトル成分に分解される。DCTの逆変換(iDCT)は数2によって表される。

【0003】

【数1】

$$F = A \cdot f \quad \dots \text{ (数4) }$$

【0008】

【数5】

$$f = A^{-1} \cdot F = A^T \cdot F \quad \dots \text{ (数5) }$$

【0009】

【数6】

$$\begin{aligned}
 & \text{f} = \begin{bmatrix} \text{f}(0) \\ \text{f}(1) \\ \dots \\ \text{f}(7) \end{bmatrix} \quad \dots \text{ (数6)} \\
 & \text{F} = \begin{bmatrix} \text{F}(0) \\ \text{F}(1) \\ \dots \\ \text{F}(7) \end{bmatrix} \quad \dots \text{ (数7)} \\
 & \text{A}^T = \begin{bmatrix} 0.3536 & 0.4904 & 0.4619 & 0.4157 & 0.3536 & 0.2778 & 0.1913 & 0.0975 \\ 0.3536 & 0.4157 & 0.1913 & -0.0975 & -0.3536 & -0.4904 & -0.4619 & -0.2778 \\ 0.3536 & 0.2778 & -0.1913 & -0.4904 & -0.3536 & 0.0975 & 0.4619 & 0.4157 \\ 0.3536 & 0.0975 & -0.4619 & -0.2778 & 0.3536 & 0.4157 & -0.1913 & -0.4904 \\ 0.3536 & -0.0975 & -0.4619 & 0.2778 & 0.3536 & -0.4157 & -0.1913 & 0.4904 \\ 0.3536 & -0.2778 & -0.1913 & 0.4904 & -0.3536 & -0.0975 & 0.4619 & -0.4157 \\ 0.3536 & -0.4157 & 0.1913 & 0.0975 & -0.3536 & 0.4904 & -0.4619 & 0.2778 \\ 0.3536 & -0.4904 & 0.4619 & -0.4157 & 0.3536 & -0.2778 & 0.1913 & -0.0975 \end{bmatrix} \quad \dots \text{ (数8)}
 \end{aligned}$$

【0012】DCTは主に画像信号の符号化に用いられる。画像信号に用いられるDCTは数1、数2に示した8次のものが多く、実際には、数8に示されるように、水平方向・垂直方向に対し変換を施す2次元のものが主に用いられる。即ち水平8画素、垂直8画素の64画素のブロックに対す64個のそれぞれ直交する周波数成分に分解する。逆変換(iDCT)は周波数成分を元の画像に変換する手法である。

【0013】数9では64画素の変換に4096回の積※

$$\text{f}(x,y) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 \text{C}(u)\text{C}(v)\text{F}(u,v) \cos \left\{ \frac{(2x+1)u\pi}{16} \right\} \cos \left\{ \frac{(2y+1)v\pi}{16} \right\} \quad \dots \text{ (数9)}$$

【0015】

【数10】

$$\text{f} = \text{A}^T \cdot \text{F} \cdot \text{A} \quad \dots \text{ (数10)}$$

【0016】これらのことから、以下、説明を簡単にするため、主に1次元のiDCTを中心に説明を行う。

【0017】図1、図2に一般的なiDCT回路6の構成を示す。図1、図2は数5をもとに回路化した例である。直交変換においては入力されるデータ数と変換後のデータ数は同じである。そのため、iDCT回路の入出力条件としては、入力データレートと出力データレートは同じで、しかも間断なくデータを入力できることが望まれている。そのため、図1、図2の回路では数8の各行の演算を並列に行っている。

【0018】図1では入力された信号1に対し、八つの積和演算器2-1～8で数2の積和演算を実行し、復号画像3-1～8(それぞれf(0)～f(7)に対応)を得る。計算結果はブロックの計算終了後、速やかにそれぞれ記憶回路7-1～8に記憶され、積和演算器2-1～8は次のブロックのデータの処理を実行する。記憶回路7-1～8の信号8-1～8は選択器4で順に選択

※和演算が必要であるが、一般には数10のように1次元の行列演算の形に変形することにより、積和演算の数を1024回に減らすとともに数4から数8の1次元のDCT、iDCTに帰着させることが出来る。2次元のDCT、iDCTの高速化、簡略化は1次元のDCT、iDCTの高速化、簡略化を行うことにより実現が可能である。

【0014】

【数9】

30 され出力画像信号5を得る。

【0019】積和演算回路2は図2に示される構成をしている。入力信号1は制御回路13の出力する係数信号14と積算器10で積算される。積算器10の出力結果は記憶回路12の内容18と加算器11において加算され、再び記憶回路12に記憶される。ただし、各ブロックの第一のデータは加算器11において加算処理されず、直接記憶回路12に入力して初期化を行う。これらの積算・加算処理を入力データに同期して8回行うことにより数1の計算を行うことができる。係数信号14は数8の行列の各行に対応する。すなわち、積和演算回路2-1では数8の1行目の係数が第一列目より順次用いられ、積和演算回路2-2においては2行目の係数が、以下同様に2-nの回路ではn行目の係数が用いられる。

【0020】図3は上記で説明した積和演算回路2の動作タイミングチャートの形で示す。

【0021】DCTの変換は数8を転置した行列を用いることにより上記で説明した逆変換と同じ回路により実行することができる。

50 【0022】また、2次元のiDCT(8×8)は図4

に示すように1次元i DCT回路6を継続に接続することにより実現できる。2次元の入力信号は水平成分を優先してスキャンし、64個の1次元信号54として第一の逆変換回路2-1に入力される。逆変換回路2-1は64個の入力信号を8個ずつ処理して行き、結果を64信号分のメモリ50に格納する。次にメモリ50に格納された64個の信号を2次元に並べ直し、さらに転置した後に再び1次元にスキャンして、第二の逆変換回路に入力する。このメモリ50のデータ操作はメモリの読み書きのアドレスを変えることによって容易に実行することができる。

【0023】即ち、入力は順にアドレス0, 1, 2, ..., 8, 9, ..., 63と書き込み、出力は0, 8, 16, 32, ..., 1, 9, ..., 63の順に読み出せばよい。第二の逆変換の結果復号された信号55が得られる。入力を間断なく行うために、スイッチ52, 50およびメモリ51を付加し、2回の逆変換操作をパイプライン処理を行う。逆変換回路2-1の結果はスイッチ52を介してメモリ50に書き込まれている時に、逆変換回路2-2ではメモリ51にある1操作前の結果をスイッチ51を介して読み出す。一巡の操作が終了した時点でスイッチ52, 53の接続先を変更することによりパイプライン処理を行うことができる。

【0024】これらのi DCTあるいはDCT回路2の中の積和演算回路の数は、積和演算を入力データの8倍で動作させることにより、一つに減らすことも可能である。

【0025】

【発明が解決しようとする課題】DCTおよびi DCTの計算は、八つの積和演算回路を用いることによりデータを間断なく処理する事ができるが、積算器の回路規模は非常に大きいため、DCT回路あるいはi DCT回路全体の回路規模が増大してしまう。一方、一つの積和演算回路のみで実行しようとする従来の8倍の動作速度を必要とし、その実現が難しい。

【0026】

【課題を解決するための手段】上記の課題を解決するには、積算に用いるデータを1ビットデータに分割し、積\*

\*算部分をシフト演算に置き換え、積和演算後に各ビットのデータを加算することによって実現できる。

【0027】

【作用】上記の手段により従来と同じ動作速度で、しかも、回路規模も増加させることなしに変換あるいは逆変換処理を実行することができる。

【0028】

【実施例】図5の実施例を用いて本発明の説明を行う。図5の実施例は図1と同じ動作を行うものである。図5中25の部分が本発明の部分である。その詳細を図6および図7に示す。

【0029】図5の演算のアルゴリズムを数11~18を用いて説明する。本発明では実数である行列の係数と入力信号の積算の方法が主な特徴となる。例えば、-1.306...の係数との積算を考えると、その絶対値の2進数表示は1.010011...となる。これを、

$$\begin{aligned} &1.000000 \\ &+0.010000 \\ &+0.000010 \\ &+0.000001 \end{aligned}$$

の成分に分割した後に積算を実行する。各数値は最大一つの“1”のビットを有しているため、積算はシフト演算に置き換えることができる。入力信号をFとしたとき、

$$F * -1.010011$$

を計算する代わりに、

$$\begin{aligned} &-F * 1.000000 - F * 0.010000 \\ &- F * 0.000010 - F * 0.000001 \end{aligned}$$

を計算する。【0030】数11~数13は数8の8×8の行列を7ビット精度の2進数表現した例である。

【0031】

【数11】

$$A = \frac{1}{2\sqrt{2}}(A_L A_R) \quad \dots \text{ (数11)}$$

【0032】

【数12】

$$A_L = \begin{bmatrix} 1.000000 & 1.011000 & 1.010011 & 1.001011 \\ 1.000000 & 1.001011 & 0.100010 & -0.010001 \\ 1.000000 & 0.110010 & -0.100010 & -1.011000 \\ 1.000000 & 0.010001 & -1.010011 & -0.110010 \\ 1.000000 & -0.010001 & -1.010011 & 0.110010 \\ 1.000000 & -0.110010 & -0.100010 & 1.011000 \\ 1.000000 & -1.001011 & 0.100010 & 0.010001 \\ 1.000000 & -1.011000 & 1.010011 & -1.001011 \end{bmatrix} \quad \dots \text{ (数12)}$$

【0033】

【数13】

$$A_R = \begin{bmatrix} 1.000000 & 0.110010 & 0.100010 & 0.010001 \\ -1.000000 & -1.011000 & -1.010011 & -0.110010 \\ -1.000000 & 0.010001 & 1.010011 & 1.001011 \\ 1.000000 & 1.001011 & -0.100010 & -1.011000 \\ 1.000000 & -1.001011 & -0.100010 & 1.011000 \\ -1.000000 & -0.010001 & 1.010011 & -1.001011 \\ -1.000000 & 1.011000 & -1.010011 & 0.110010 \\ 1.000000 & -0.110010 & 0.100010 & -0.010001 \end{bmatrix} \quad \dots \text{(数13)}$$

【0034】本発明では数11～数13の行列を、行列の各要素の中に有効なビット、即ち“1”であるビットが一つ以下になるように行列を分解している。数14～数18は数12を分解した様子を示している。数12の行列の各要素の絶対値を左のビット(Most Significant Bit: MSB)より調べ、最初の“1”の部分のみを取り出した行列が数15である。次に、数12より数15を引き同様にMSBよりビットを調べ“1”の部分を取り出したものが数16の行列である。数16の行列は数12の行列の各要素のMSBより数えて2番目の“1”であるビットを示している。これらの操作を繰り返して\*

\*ゆき、数17、数18を得る。数18以降は行列の要素が全て0になるため行列の分解を中止する。この結果、数12の行列は数14のように四つの行列に分解される。

【0035】

【数14】

$$A_L = A_{L1} + A_{L2} + A_{L3} + A_{L4} \quad \dots \text{(数14)}$$

【0036】

【数15】

$$A_{L1} = \begin{bmatrix} 1.000000 & 1.000000 & 1.000000 & 1.000000 \\ 1.000000 & 1.000000 & 0.100000 & -0.010000 \\ 1.000000 & 0.100000 & -0.100000 & -1.000000 \\ 1.000000 & 0.010000 & -1.000000 & -0.100000 \\ 1.000000 & -0.010000 & -1.000000 & 0.100000 \\ 1.000000 & -0.100000 & -0.100000 & 1.000000 \\ 1.000000 & -1.000000 & 0.100000 & 0.010000 \\ 1.000000 & -1.000000 & 1.000000 & -1.000000 \end{bmatrix} \quad \dots \text{(数15)}$$

【0037】

※ ※ 【数16】

$$A_{L2} = \begin{bmatrix} 0.000000 & 0.010000 & 0.010000 & 0.001000 \\ 0.000000 & 0.001000 & 0.000010 & -0.000001 \\ 0.000000 & 0.010000 & -0.000010 & -0.010000 \\ 0.000000 & 0.000001 & -0.010000 & -0.010000 \\ 0.000000 & -0.000001 & -0.010000 & 0.010000 \\ 0.000000 & -0.010000 & -0.000010 & 0.010000 \\ 0.000000 & -0.001000 & 0.000010 & 0.000001 \\ 0.000000 & -0.010000 & 0.010000 & -0.001000 \end{bmatrix} \quad \dots \text{(数16)}$$

【0038】

★ ★ 【数17】

$$A_{L3} = \begin{bmatrix} 0.000000 & 0.001000 & 0.000010 & 0.000010 \\ 0.000000 & 0.000010 & 0.000000 & -0.000000 \\ 0.000000 & 0.000010 & -0.000000 & -0.001000 \\ 0.000000 & 0.000000 & -0.000010 & -0.000010 \\ 0.000000 & -0.000000 & -0.000010 & 0.000010 \\ 0.000000 & -0.000010 & -0.000000 & 0.001000 \\ 0.000000 & -0.000010 & 0.000000 & 0.000000 \\ 0.000000 & -0.001000 & 0.000010 & -0.000010 \end{bmatrix} \quad \dots \text{(数17)}$$

【0039】

☆ ☆ 【数18】

$$A_{L4} = \begin{bmatrix} 0.000000 & 0.000000 & 0.000001 & 0.000001 \\ 0.000000 & 0.000001 & 0.000000 & -0.000000 \\ 0.000000 & 0.000000 & -0.000000 & -0.000000 \\ 0.000000 & 0.000000 & -0.000001 & -0.000010 \\ 0.000000 & -0.000000 & -0.000001 & 0.000010 \\ 0.000000 & -0.000000 & -0.000000 & 0.000000 \\ 0.000000 & -0.000001 & 0.000000 & 0.000000 \\ 0.000000 & -0.000000 & 0.000001 & -0.000001 \end{bmatrix} \quad \dots \text{(数18)}$$

【0040】図5ではこれら数15～数18の各行列に 50 対して一つの1ビットのiDC T回路25-1～4が割

り当てられる。入力信号1は各ビット毎に分解された行列毎に演算されそれぞれの結果26-1~4は加算器63で加算され、最終結果を得る。

【0041】図6は図5の1ビットのiDCTの計算回路である。信号の流れは図1のiDCT回路と同様であるが、積和演算の代わりにシフト加算回路20-1~8が用いられている点、選択回路4の後段に定数倍回路21の回路が挿入されている点が特徴である。

【0042】図7は図6におけるシフト加算回路20の詳細図である。入力された信号1はシフト回路30において指定されたビット数nだけ右シフト、即ち2のn乗で除算される。シフト結果15は加減算回路31において過去の蓄積結果18に対し加算、あるいは減算され、記憶回路12に保持される。これらの処理を8回繰り返すことにより1ビットiDCT回路25の処理を行う。

【0043】シフト回路30におけるシフト数33や、加減算器31の加減算選択信号34は制御回路32により発生される。これらの内容およびタイミングの例を図8に示す。図8は数15との演算を行う回路の例であ

\* する。シフト数33や加減算選択34は八つのシフト加算回路においてもそれぞれ異なる。

【0044】例えば、シフト加算回路2-1では図8に示すように、1番目の入力信号に対しては0、2番目に対しては0、以下順に、0、0、0、1、1、2が出力される。シフト数33が0の時は入力信号を0ビット右シフト、即ち、数15における1.000000を掛け、1の場合は1ビット右シフト、即ち0.100000を掛けた値を出力する。一方、加減算選択信号34は図8の右のように加算(+)か減算(-)かあるいは出力をゼロにするか(0)を選択する。

【0045】これら、シフト加算した結果は記憶回路7-1~8に保持された後、選択回路4により順次選択される。選択された信号23は、その選択した順に応じた出力係数が掛けられ出力される。各記憶回路7-1~8に対応する出力係数をa0~a7としたときの演算行列を数19に示す。

【0046】

【数19】

$$A = \begin{bmatrix} a0 & a0 & a0 & a0 & a0 & a0/2 & a0/2 & a0/4 \\ a1 & a1 & a1/2 & -a1/4 & -a1 & -a1 & -a1 & -a1/2 \\ a2 & a2/2 & -a2/2 & -a2 & -a2 & a2/4 & a2 & a2 \\ a3 & a3/4 & -a3 & -a3/2 & a3 & a3 & -a3/2 & -a3 \\ a4 & -a4/4 & -a4 & a4/2 & a4 & -a4 & -a4/2 & a4 \\ a5 & -a5/2 & -a5/2 & a5 & -a5 & -a5/4 & a5 & -a5 \\ a6 & -a6 & a6/2 & a6/4 & -a6 & a6 & -a6 & a6/2 \\ a7 & -a7 & a7 & -a7 & a7 & -a7/2 & a7/2 & -a7/4 \end{bmatrix} \dots (\text{数19})$$

【0047】各出力係数a0~a7は全て同じ値でもよい。例えば、数11の場合はa0=a1=a2=a3=a4=a5=a6=a7=0.3536...(1/2/sqrt(2))の場合誤差が最も小さくなる。また、予め数15~数18の行列の代わりに数8を2進数表示し、ビット毎に分解した行列を用いることにより、a0=a1=...a7=1として定数倍回路21を削除する事も可能であり、本発明に含まれる。また、2次元iDCTの場合は先の係数0.3536を2回掛けることになるため、これらを纏めて1回の積算にすることも本発明に含まれる。即ち、1段目のiDCT回路の定数倍回路21を削除し、2段目のiDCT回路において2回分の定数倍を行う。先の例の場合には、0.3536...の2乗は0.125となるため定数倍回路は、3ビット右シフト回路に置き換えることができる。

【0048】上記の行列の分解においては最大一つの“1”であるビットを持つ要素に分解し、計算した後に、加算あるいは減算を行ったが、加算と減算を組み合わせることによって計算回数を削減することが可能である。例えば、行列の係数が、2進数表示で0.11111111の時には、上記で説明した方法では6回のシフトと加算が必要であった。これを1.000000-0.000001と考えることにより2回のシフトと加減算で実行できる。このような最適化を行った場合も本発明に

包含される。

【0049】図7において係数が0である部分は計算を行わなくてもよい。例えば、一つの行の係数がすべて0であるような場合には図5においてその行に対するシフト加算回路20を省略できる。また、一つの行の中で0でない係数が一つの場合には図7の加減算回路を省略することができる。このような最適化を行った場合も本発明に含まれる。また、各1ビットiDCT回路において最適化を行った結果、図5の25-1~4のそれぞれの回路構成が異なってしまう場合も本発明に含まれる。

【0050】図9は本発明の第二の実施例である。図9は図5の実施例の1ビットiDCT回路25-2~4の部分の纏めてiDCT回路6に置き換えた点が特徴である。また、計算順も1ビットiDCT25回路とiDCT回路6が同時に計算するのではなく1ビットiDCT回路25が先に計算を行い、計算終了した後の任意のタイミングにiDCT回路6が計算を開始することができる。1ビットiDCT回路25が先に近似計算を行い、その結果を信号線67、スイッチ65を介して出力し、後にiDCT回路6において誤差信号を計算して先の近似値に加算し、正確な値をメモリ64、スイッチ65を介して出力する。

【0051】1ビットiDCTの計算に用いる行列は図

11

10に従い発生する。これは数20の行列を用いて計算を行っていることになる。i DCT回路6では数8の行列の各要素から数20の行列の各対応する要素の値を引いた値を要素とする行列を用いて演算を行う。これらの処理は特にソフトウェアなどを用いて順次にi DCTの計算を行う場合に有効である。まず、先に1ビットi DCTにより高速に近似計算を行い、次に正確な計算を行\*

$$A = \begin{bmatrix} a0 & a0 & a0 & a0 & a0 & a0/2 & a0/2 & 0 \\ a1 & a1 & a1/2 & 0 & -a1 & -a1 & -a1 & -a1/2 \\ a2 & a2/2 & -a2/2 & -a2 & -a2 & 0 & a2 & a2 \\ a3 & 0 & -a3 & -a3/2 & a3 & a3 & -a3/2 & -a3 \\ a4 & 0 & -a4 & a4/2 & a4 & -a4 & -a4/2 & a4 \\ a5 & -a5/2 & -a5/2 & a5 & -a5 & 0 & a5 & -a5 \\ a6 & -a6 & a6/2 & 0 & -a6 & a6 & -a6 & a6/2 \\ a7 & -a7 & a7 & -a7 & a7 & -a7/2 & a7/2 & 0 \end{bmatrix} \dots (\text{数 } 20)$$

【0053】なお、図9において1ビットi DCT回路25は一つであるが、これを複数個用いて計算をし、その誤差をi DCT回路6で計算することも本発明に包含される。例えば、図5の四つの1ビットi DCT回路を順次計算し、加算した後に誤差成分をi DCT回路6で計算し、それまでの計算結果に加えることにより、図5

よりもさらに計算精度の高いi DCTを行うことが可能となる。  
【0054】図10あるいは数20で示される行列は先の図8あるいは数19で示される行列を簡略化させたものである。3ビット以上シフトする部分を0に置き換え※

$$A = \begin{bmatrix} a0 & a0 & a0 & a0 & a0 & a0 & a0 & 0 \\ a1 & a1 & a1 & 0 & -a1 & -a1 & -a1 & -a1 \\ a2 & a2 & -a2 & -a2 & -a2 & 0 & a2 & a2 \\ a3 & 0 & -a3 & -a3 & a3 & a3 & -a3 & -a3 \\ a4 & 0 & -a4 & a4 & a4 & -a4 & -a4 & a4 \\ a5 & -a5 & -a5 & a5 & -a5 & 0 & a5 & -a5 \\ a6 & -a6 & a6 & 0 & -a6 & a6 & -a6 & a6 \\ a7 & -a7 & a7 & -a7 & a7 & -a7 & a7 & 0 \end{bmatrix} \dots (\text{数 } 21)$$

【0057】図5、図9等の本発明の回路はi DCTの計算回路として説明を行ったが、先にも触れたように行列部分を変えることによりDCTにも容易に適用可能である。即ち、例えば数22に示す行列を計算することによりDCTを計算することができる。先に示した実施例★

$$A = \begin{bmatrix} a0 & a0 & a0 & a0 & a0 & a0 & a0 & a0 \\ a1 & a1 & a1/2 & a1/4 & -a1/4 & -a1/2 & -a1 & -a1 \\ a2 & a2/2 & -a2/2 & -a2 & -a2 & -a2/2 & a2/2 & a2 \\ a3 & -a3/4 & -a3 & -a3/2 & a3/2 & a3 & a3/4 & -a3 \\ a4 & -a4 & -a4 & a4 & a4 & -a4 & -a4 & a4 \\ a5/2 & -a5 & a5/4 & a5 & -a5 & -a5/4 & a5 & -a5/2 \\ a6/2 & -a6 & a6/2 & -a6/2 & -a6/2 & a6 & -a6 & a6/2 \\ a7/4 & -a7/2 & a7 & -a7 & a7 & -a7 & a7/2 & -a7/4 \end{bmatrix} \dots (\text{数 } 22)$$

【0059】図12は図5の第三の実施例である。一つの1ビットi DCT回路25をその行列の係数を変えることにより、図5の1ビットi DCT回路25-1~4の計算を順次行っていく。それらの結果66をメモリ64に蓄積加算することによって図5と同じ結果を得ることができる。

12

\*う。これにより、例えば、画像の復号化の処理では、処理開始直後にやや劣化のある画像が表示され、次第に劣化のない画像に置き換わって行くといった階層的な表示が可能である。

【0052】

【数20】

※ることによって計算量を削減している。

【0055】図11は図7の回路の変形例であり、図9の1ビットi DCT部分に適用することにより効果的になる。図11は図7の回路からシフト回路30を削除したものである。これにより、例えば演算に使用する行列は数21のようになり、計算誤差は数19、数20に比べやや大きくなるが、ソフトウェア処理では高速化が、ハードウェア処理では回路の簡略化を図ることができる。

【0056】

【数21】

★および、変形例は全てDCTの計算にも適用が可能である。

【0058】

【数22】

【0060】図13は第四の実施例であり、図9の変形例である。図9においては1ビットi DCT25の計算結果をi DCT6の結果に加算していたが、図13の実施例ではi DCT回路において正確なi DCTを行い、結果の加算を行わない。これにより図9より少ない計算量で、図9と同じ効果を得ることができる。



【0061】以上説明したi DCTおよびDCTの演算は各実施例そのものでなく、組み合わせた形で実現できる。例えば、図5の実施例の1ビットi DCT回路25-1に数21の行列を組み合わせることも可能である。

【0062】図5において1ビットi DCT回路には高速演算が適用可能である。1ビットi DCT回路に高速演算を用いた場合も本発明に包含される。例えば、図14は数21の係数を用いる時の1ビットi DCT回路の高速演算である。入力信号F(0)~F(7)は矢印で示されたデータどうし加算あるいは減算され丸印で示される中間結果、例えばF02p等、になる。これら中間結果もさらに加算あるいは減算され最終的にf(0)~f(7)を得る。

【0063】図15は図14の高速演算を行うソフトウェアの例である。C言語で書かれたプログラムの一部を示している。全ての変数は整数型で、名前は図14の中間結果の名前に対応している。

【0064】図14の高速演算により、1次元のi DCTは加減算24回と、定数倍の積算8回になる。また、2次元のi DCTを行っても加減算48回と、3ビットのシフトを64回行うだけで実行できる。

【0065】図16、図17は本発明を用いた1次元/2次元のDCT回路およびi DCT回路のブロック図である。2次元DCT、i DCTは数10あるいは図4に示しているものと同一であり、2次元DCTの結果72と中間結果である1次元のDCTの結果71をスイッチ70で切り替えている。また、図17のi DCTではスイッチ73にて2次元i DCTの入力75と1次元i DCTの入力74を振り分け、1次元i DCTの入力は2\*30

	DCT		i DCT		
	6-3	6-4	6-1	6-2	1次元DCTの出力倍率
A	1/2/sq2	1/2/sq2	1/2/sq2	1/2/sq2	1
B	1	1/8	1	1/8	2*sq2
C	1/2	1/4	1/2	1/4	sq2
D	1/4	1/2	1/4	1/2	1/sq2
E	1/8	1	1/8	1	1/2/sq2

上記のB、C、D、Eが本発明に包含されることは明白である。またB、C、D、Eは積和演算回路を用いた1次元/2次元DCT、i DCT回路にも適用可能である。

【0067】図16において1次元DCT出力71と2次元DCT出力はスイッチ70により切り替えられているが、これらを同時に出力することも本発明に包含される。図17において入力切替スイッチ73は入力直後に入っているが、スイッチ52の直前に配置し、1段目のi DCT回路6-1の結果と入力された信号とを切り替えてもよい。また、スイッチ73は2段目のi DCT回路6-2の直前に配置しても同様の効果が得られる。さらに、図16と同様な構成をとり、1段目のi DCTの

\* 段目のi DCT回路6-2に入力している。なお、図16のDCT回路6-3、4とi DCT回路6-1、2は先に述べたように同じ回路で係数のみを変えることによって実現できる。図16、図17において問題になるのはDCT回路6-3、4、i DCT6-1、2の出力係数a0~a7の値である。1次元のDCT、i DCT回路ではa0=a1=...a7=0.3536... (=1/2/sqrt(2): sqrtは平方根を表す) である。しかし、行列の係数に実数を用いると直流成分のF(0)が実数値となり、計算精度によって誤差が生じやすくなり、復号化した画質にも誤差が見えやすくなる。そこで、1次元のDCTの結果に予め2\*sqrt(2) (=2.8284...) あるいはsqrt(2) (=1.4142...) あるいは1/sqrt(2) (=0.7071...) あるいは1/2/sqrt(2) (=0.3535...) を乗じておき、逆にi DCTの時にこれらの値を除することによって、直流成分の計算が整数演算で実現でき、画質に最も影響を及ぼす直流成分の誤差をなくすることができる。以上をまとめると下のようになる。下の表は1次元DCTの結果を予め先の3通りの倍率にするとときの各DCT回路6-3、6-4およびi DCT回路6-1、6-2における出力係数a0~a7の値(全て同じ値の場合)を示したものである。なお、表中sq2はsqrt(2)、即ち、1.4142...を表す。Aの項が数1、数2の定義通り、B、C、D、Eがそれぞれ2\*sqrt(2)倍、sqrt(2)倍、1/sqrt(2)倍、1/2/sqrt(2)倍に相当する。

【0066】

【表1】

結果を1次元のi DCTの結果として出力してもよい。以上の変形例はすべて本発明に包含される。

【0068】図16あるいは図17において信号を入力してから1次元DCTあるいはi DCTの結果が出るまでの時間と、2次元のDCT、i DCTが出るまでの時間は異なる。1次元のDCTあるいはi DCTの結果を遅延回路において遅延させて、2次元のDCTあるいはi DCTの結果と同じタイミングに出力することができる。あるいは、1次元のDCTあるいはi DCTの入力を遅延回路において遅延させることによって同様の効果が得られる。これらの遅延回路は入力信号の入力順あるいは出力信号の出力順を変えるバッファと兼ねることも可能である。上記で述べた遅延回路を含む1次元/2次元

DCTあるいはiDCT回路も本発明に包含される。

【0069】

【発明の効果】直交変換の行列演算を1ビット毎に計算することにより従来と同じ動作速度で、しかも、回路規模も増加させることなしに変換あるいは逆変換処理を実行することができる。さらに、各ビットの計算をMSBより順次行うことにより、ソフトウェア処理などの場合に概算値を先に高速で計算し、正確な値を順次計算することも可能になる。

【図面の簡単な説明】

【図1】従来の1次元iDCT回路のブロック図。

【図2】図1中の積和演算回路のブロック図。

【図3】図2のiDCT回路の動作タイミングチャート。

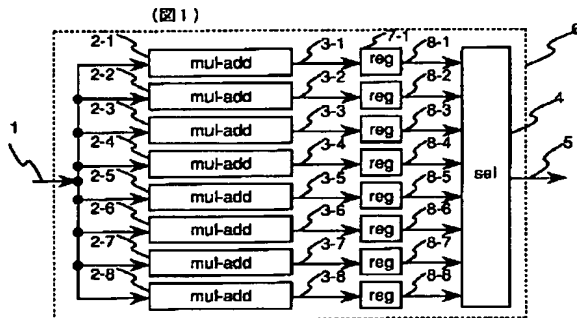
【図4】2次元iDCTの構成例のブロック図。

【図5】本発明による1次元iDCT回路のブロック図。

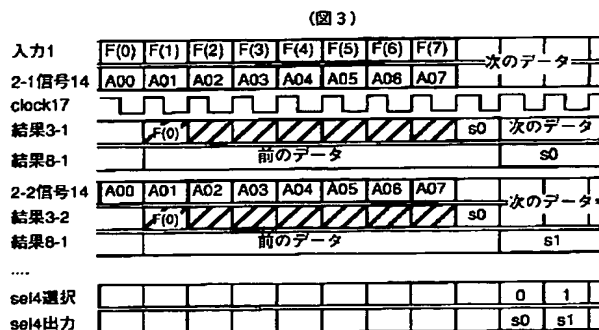
【図6】図5の1ビットiDCT部分のブロック図。

【図7】図6のシフト加算回路のブロック図。

【図1】



【図3】



\*【図8】図6における制御信号の説明図。

【図9】本発明の第二の実施例のブロック図。

【図10】図6における制御信号の第二の実施例の説明図。

【図11】図7のシフト加算回路のブロック図。

【図12】本発明の第三の実施例のブロック図。

【図13】本発明の第四の実施例のブロック図。

【図14】図11の高速演算の信号フローチャート。

【図15】図13をソフトウェアで実現した例の説明図。

10 図。

【図16】1次元のDCTと2次元のDCTを一つの回路で実現するブロック図。

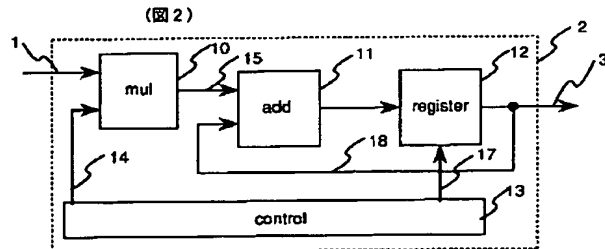
【図17】1次元のiDCTと2次元のiDCTを一つの回路で実現するブロック図。

【符号の説明】

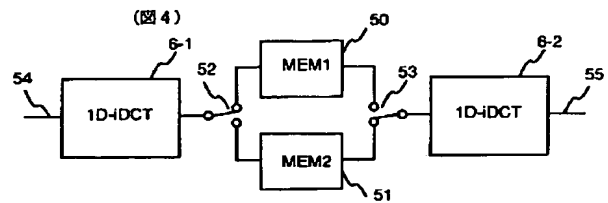
1…入力データ、2…積和演算回路、4…データ選択回路、5…出力データ、6…1次元iDCT回路、10…積算器、11…加算器、25…1次元1ビットiDCT回路、30…シフト回路、31…加減算回路。

\*

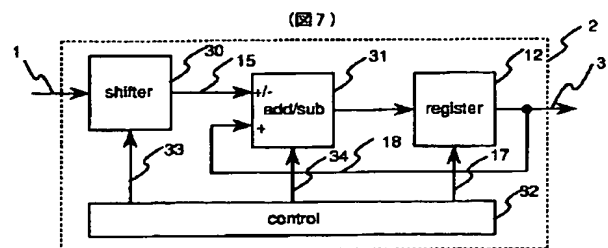
【図2】



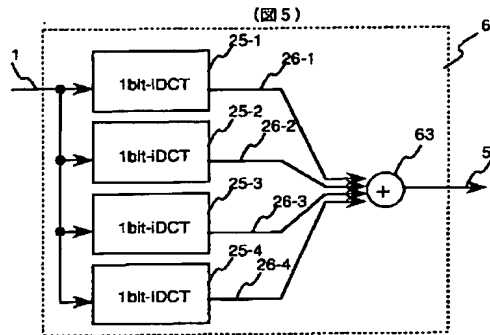
【図4】



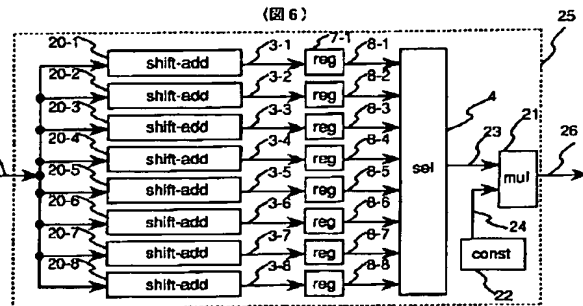
【図7】



【図5】



【図6】

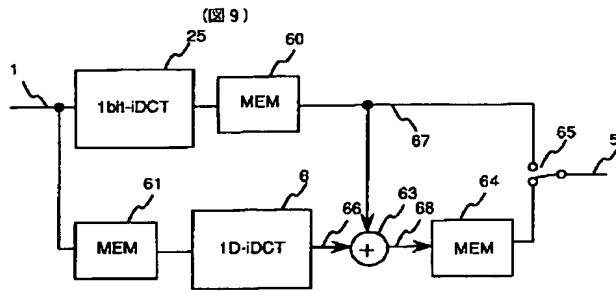


【図8】

(図8)

入力1:	0	1	2	3	4	5	6	7	入力1:	2	1	2	3	4	5	6	7
2-1の信号33:	0	0	0	0	1	1	2		2-1の信号34:	+	+	+	+	+	+	+	+
2-2の信号33:	0	0	1	2	0	0	1		2-2の信号34:	+	+	+	+	+	+	+	+
2-3の信号33:	0	1	1	0	0	2	0		2-3の信号34:	+	+	+	+	+	+	+	+
2-4の信号33:	0	2	0	1	0	0	1		2-4の信号34:	+	+	+	+	+	+	+	+
2-5の信号33:	0	2	0	1	0	0	1		2-5の信号34:	+	+	+	+	+	+	+	+
2-6の信号33:	0	1	1	0	0	2	0		2-6の信号34:	+	+	+	+	+	+	+	+
2-7の信号33:	0	0	1	2	0	0	1		2-7の信号34:	+	+	+	+	+	+	+	+
2-8の信号33:	0	0	0	0	1	1	2		2-8の信号34:	+	+	+	+	+	+	+	+

【図9】

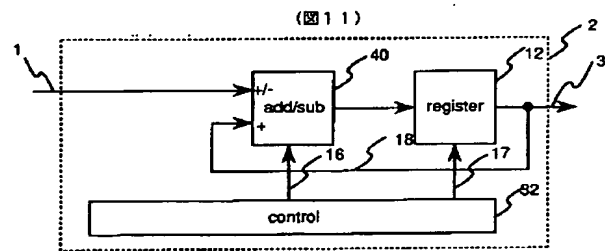


【図10】

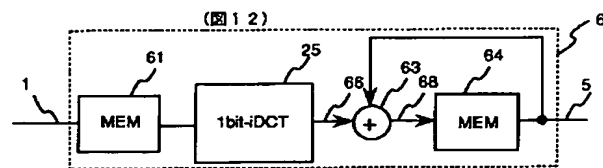
(図10)

入力1:	0	1	2	3	4	5	6	7	入力1:	0	1	2	3	4	5	6	7
2-1の信号33:	0	0	0	0	1	1	2		2-1の信号34:	+	+	+	+	+	+	+	+
2-2の信号33:	0	0	1	2	0	0	1		2-2の信号34:	+	+	+	+	+	+	+	+
2-3の信号33:	0	1	1	0	0	2	0		2-3の信号34:	+	+	+	+	+	+	+	+
2-4の信号33:	0	2	0	1	0	0	1		2-4の信号34:	+	+	+	+	+	+	+	+
2-5の信号33:	0	2	0	1	0	0	1		2-5の信号34:	+	+	+	+	+	+	+	+
2-6の信号33:	0	1	1	0	0	2	0		2-6の信号34:	+	+	+	+	+	+	+	+
2-7の信号33:	0	0	1	2	0	0	1		2-7の信号34:	+	+	+	+	+	+	+	+
2-8の信号33:	0	0	0	0	1	1	2		2-8の信号34:	+	+	+	+	+	+	+	+

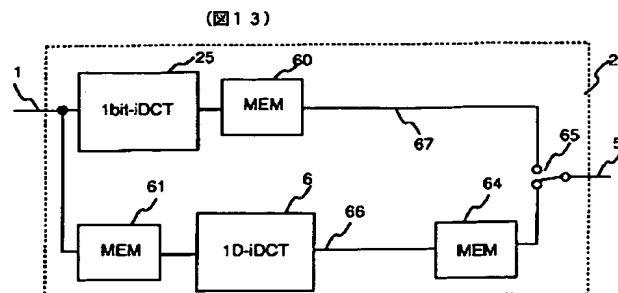
【図11】



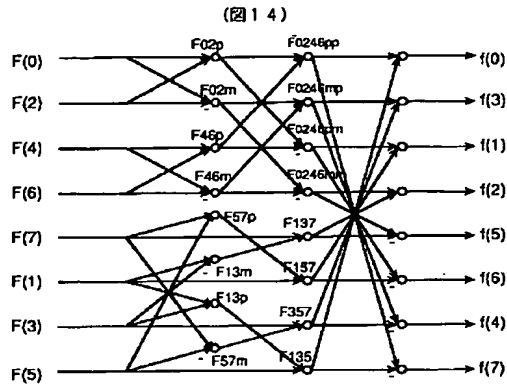
【図12】



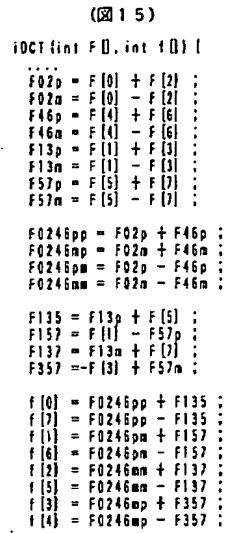
【図13】



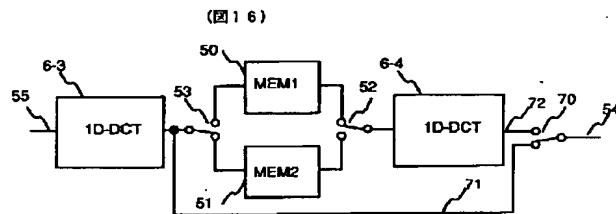
【図14】



【図15】



【図16】



【図17】

